Chef による Zabbix 監視自動化について+α

株式会社 サイバーエージェント
Ameba Infra. Unit

長谷川　誠

@Zabbix Conference Japan 2013

アジェンダ

- はじめに
- Chef とは何か
- Zabbix Server のセットアップ
- Zabbix Agent のセットアップ
- Zabbix Serverとの連携
- まとめ
- おわりに

ExtraContents

- Template の管理と deploy
- Classic style からの脱却

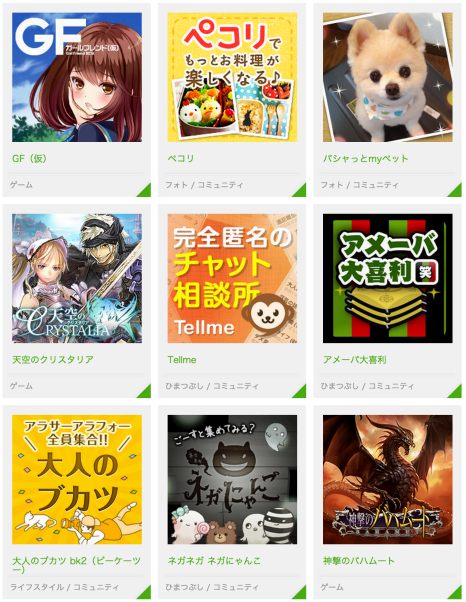
```
{
 "name": "hasegawa_makoto",
 "description": "Zabbix Conference Japan 2013",
 "json_class": "Chef::Role",
 "chef_type": "role",
 "run_list": [
   "recipe[introduction]",
   "recipe[chef::explain]",
   "recipe[zabbix::server_setup]",
   "recipe[zabbix::agent_setup]",
   "recipe[zabbix::api]",
   "recipe[summary]",
   "recipe[closing]",
 ],
 "extra_run_lists": {
   "recipe[zabbix::template_management]",
   "recipe[zabbix::custom_style]"
 }
}

# chef-solo -o zabbix::conference
```

はじめに

```
# chef-solo -o introduction
```

サーバー構築の自動化の仕組みが必要になった背景

- 次々と新しいプロジェクトが生まれていく
- 逆を言うと流行らないものは潰していく



重要なのはスピード感

- サーバー構築に時間をかけていると、それだけ開発が遅れることに
- 遅れた分は機会損失につながる

機会損失をできるだけなくしていく

- サーバーを構築するまでは既に自動化済み（chef）
- しかし構築した後の監視設定までは自動化に入っていなかった

まだ自動化できてない部分があるのであれば、自動化しちゃえばいいじゃない
また、すでにある自動化の仕組み（chef）に組み込むのが効率いい

Chef による監視自動化を構築

Why Zabbix ?

一番のポイントは API が充実していること

- Nagios (icinga) や munin や cacti でやろうとするとめんどくさい

死活監視とリソース監視に対応している

- Nagios + Munin や Nagios + Cacti といった構成を Zabbix で集約

Free / OSS

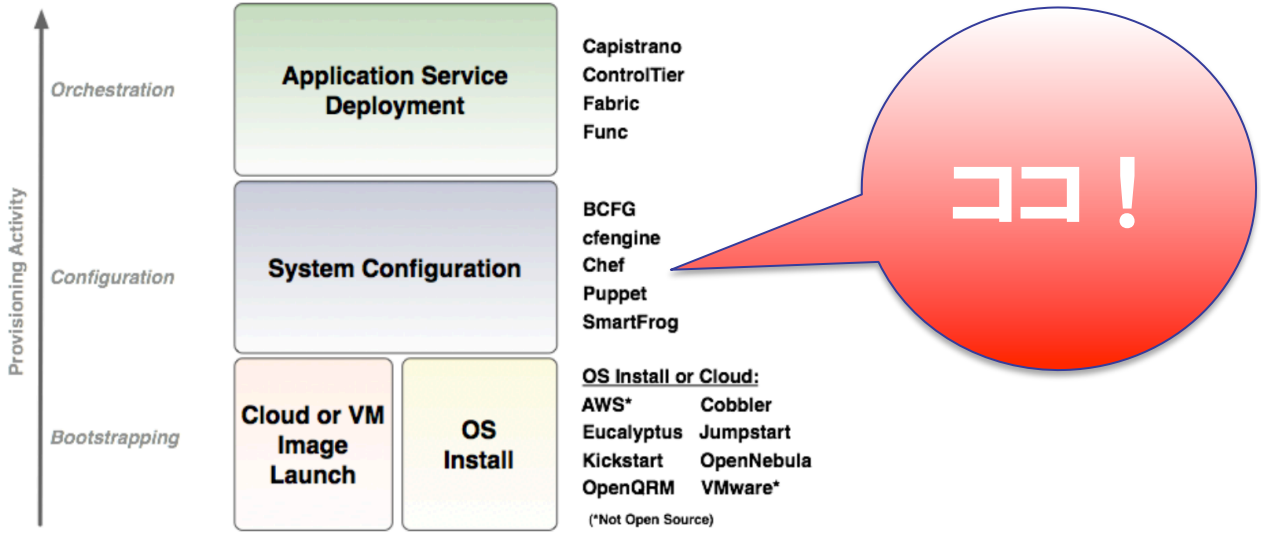
- ありがとうございます

Chef とは何か

```
# chef-solo -o chef::explain
```

サーバーを構築 (Provisioningなんてよく言われてます) する時によく出てくるお話

Bootstrapping / Configuration / Orchestration



http://dev2ops.org/toolchain/

Chefは Configuration を担当するもの

■Chef の概念

Infrastructures as Code

・プログラミングによるサーバーセットアップ

・構築手順書を見ながら手動で作業することからの脱却

冪等性（べきとうせい）

・何度やっても同じ状態にセットアップしてくれることを**ある程度**担保してくれる

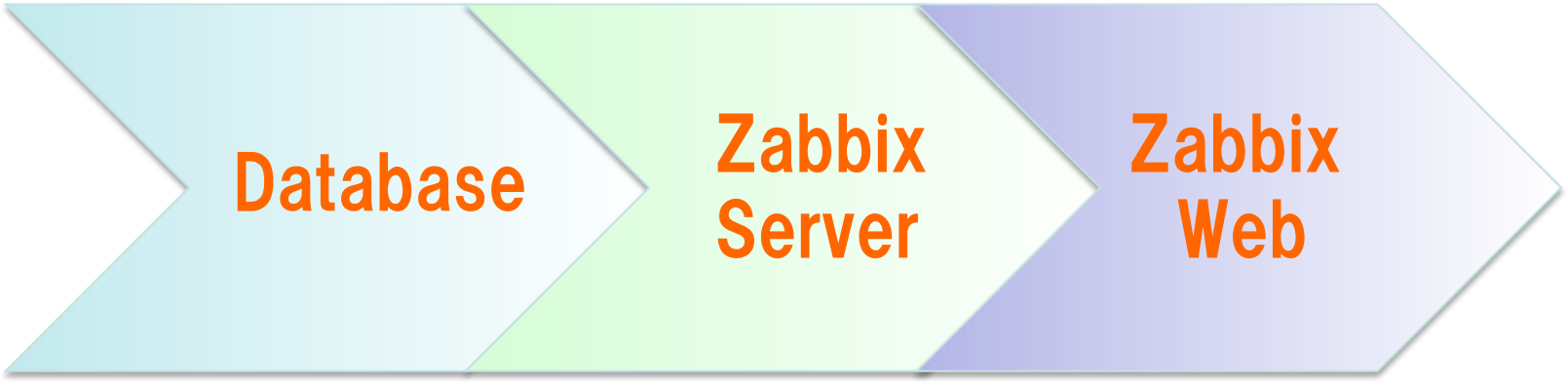
・誰がいつやっても同じクオリティでセットアップしてくれる

自動化には欠かせないもの

Zabbix Server のセットアップ

```
# chef-solo -o zabbix::server_setup
```

Zabbix Server を語る上で必要なコンポーネントは 3 つ

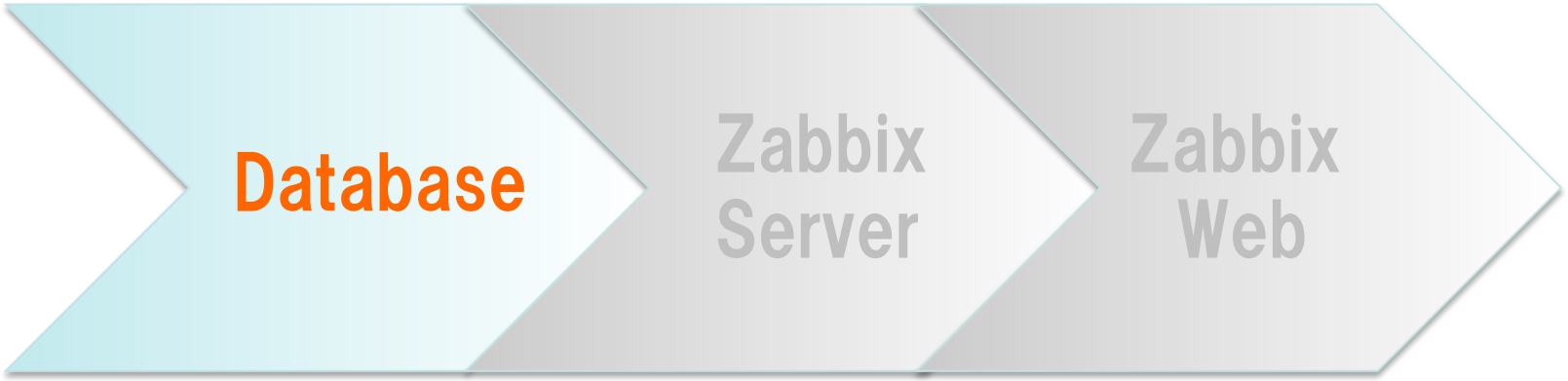


Chef の Recipe もコンポーネント毎に作成

前提条件

- 今回は Database に MySQL (5.5) を使用します

Zabbix Server を語る上で必要なコンポーネントは 3 つ



では Database（MySQL）から行きましょう

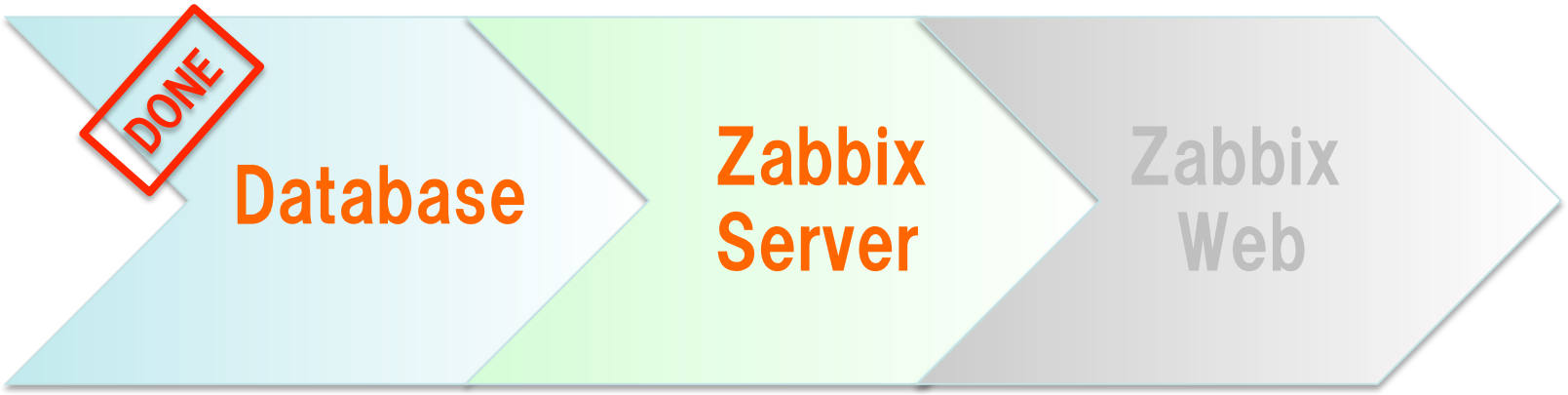
runlist [mysql::server]

1. 各種パラメーターの自動設定
   - innodb_buffer_pool_size（memory の 70%）
   - innodb_file_per_table
   - innodb_file_format = ‘Barracuda’
   - などなど

2. 上記を元に /etc/my.cnf を作成

3. パッケージインストール
   - MySQL 公式の rpm（dev.mysql.com）

Zabbix Server を語る上で必要なコンポーネントは 3 つ



Database の準備ができたところで zabbix server 行きましょう

runlist [zabbix::server]

1. パッケージインストールの前に
   - いくつかの依存パッケージをインストール
     fping (version > 3) , iksemel, snmptt

2. パッケージインストール
   - Zabbix 公式の rpm (repo.zabbix.com)

3. Database の作成
   - 次ページで詳細を書きます

4. /etc/zabbix/zabbix_server.conf の作成
   - DBまわりのパラメータ (DBHost, DBName, DBUser …) 設定

runlist [zabbix::server]

- Database 作成の詳細

・いつもの手順
- CREATE DATABASE #{db_name} CHARACTER SET utf8
- GRANT ALL PRIVILEGES ON #{db_name}.* TO #{db_user}@#{db_host} IDENTIFIED BY #{db_pass}

・いつもの手順2
- mysql -uroot zabbix < /usr/share/doc/zabbix-server-mysql-2.0.X/create/schema.sql
- mysql -uroot zabbix < /usr/share/doc/zabbix-server-mysql-2.0.X/create/images.sql
- mysql -uroot zabbix < /usr/share/doc/zabbix-server-mysql-2.0.X/create/data.sql

・Table 圧縮 （ History 系はディスクを圧迫しやすいので ）
- ALTER TABLE #{table} ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=8
  - » history
  - » history_log
  - » …
  - » events

runlist [zabbix::server]

- Database 作成の詳細

```
mysql> SELECT TABLE_SCHEMA,TABLE_NAME,CREATE_OPTIONS,ROW_FORMAT
    -> FROM information_schema.tables where ROW_FORMAT = 'Compressed' and TABLE_SCHEMA = 'zabbix' ;
+--------------+------------------+----------------------------------------+------------+
| TABLE_SCHEMA | TABLE_NAME       | CREATE_OPTIONS                         | ROW_FORMAT |
+--------------+------------------+----------------------------------------+------------+
| zabbix       | events           | row_format=COMPRESSED KEY_BLOCK_SIZE=8 | Compressed |
| zabbix       | history          | row_format=COMPRESSED KEY_BLOCK_SIZE=8 | Compressed |
| zabbix       | history_log      | row_format=COMPRESSED KEY_BLOCK_SIZE=8 | Compressed |
| zabbix       | history_str      | row_format=COMPRESSED KEY_BLOCK_SIZE=8 | Compressed |
| zabbix       | history_str_sync | row_format=COMPRESSED KEY_BLOCK_SIZE=8 | Compressed |
| zabbix       | history_sync     | row_format=COMPRESSED KEY_BLOCK_SIZE=8 | Compressed |
| zabbix       | history_text     | row_format=COMPRESSED KEY_BLOCK_SIZE=8 | Compressed |
| zabbix       | history_uint     | row_format=COMPRESSED KEY_BLOCK_SIZE=8 | Compressed |
| zabbix       | history_uint_sync | row_format=COMPRESSED KEY_BLOCK_SIZE=8 | Compressed |
+--------------+------------------+----------------------------------------+------------+
9 rows in set (0.01 sec)
```

先ほど innodb_file_format = 'Barracuda' を指定した理由はこれです

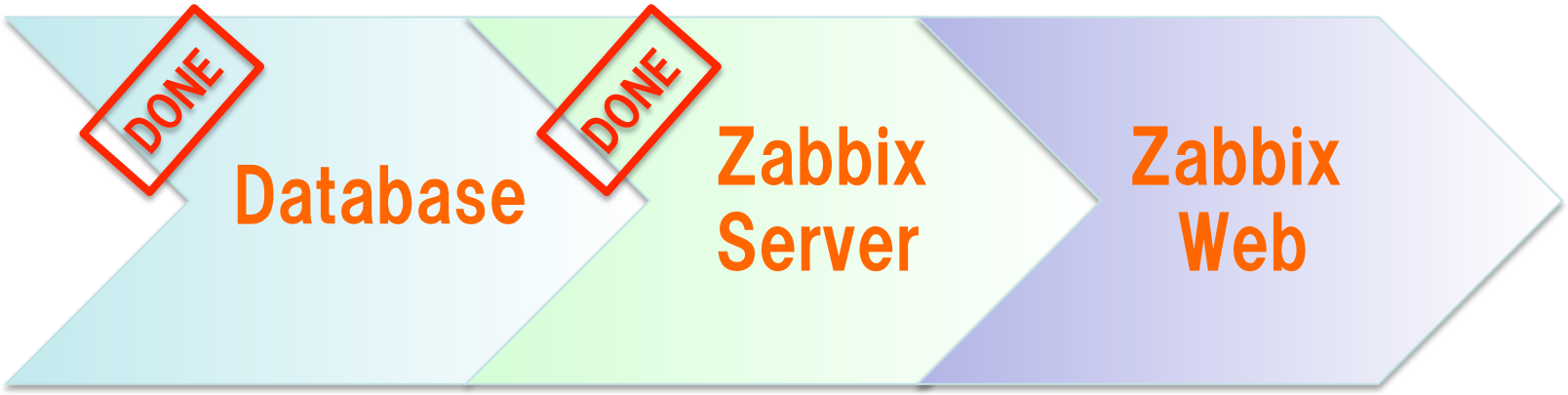
runlist [zabbix::server]

- zabbix_server.conf の作成

DB以外でデフォルト値より変更しているのは下記の設定です

- JMXまわり
  - JavaGateway, JavaGatewayPort, StartJavaPollers
- MaxHousekeeperDelete はちょっと大きくしておく
  - MaxHousekeeperDelete=1000（500）
- Cache まわり
  - CacheSize=32M（8M）
  - CacheUpdateFrequency=30（60）
  - HistoryCacheSize=1G（8M）
  - TrendCacheSize=1G（4M）
  - HistoryTextCacheSize=128M（16M）

Zabbix Server を語る上で必要なコンポーネントは 3 つ



最後に web の設定をしましょう

runlist [zabbix::web]

1. パッケージインストールの前に
   - いくつかの依存パッケージをインストール

     httpd, php
     - php.ini (or /etc/httpd/conf.d/zabbix.conf) date.timezone = "Asia/Tokyo"

2. パッケージインストール
   - Zabbix 公式の rpm (repo.zabbix.com)

3. /etc/zabbix/web/zabbix.conf.php を作成
   - 詳細は次ページ

4. API を使った初期セットアップ
   - 詳細は次次ページ

runlist [zabbix::web]

/etc/zabbix/web/zabbix.conf.php を作成しておくと・・・

最初に出てくる Welcome 画面 (setup 画面) が出てこなくなるので
最初から dashboard が出てきます

setup 画面で行う各種確認は chef で構築されていれば
確認しなくてもOKなはず

というわけで
zabbix.conf.php の作成は chef 側で作成してしまいます

runlist [zabbix::web]

API を使った初期セットアップ

生まれたままの zabbix server に対して API 経由で設定を入れます

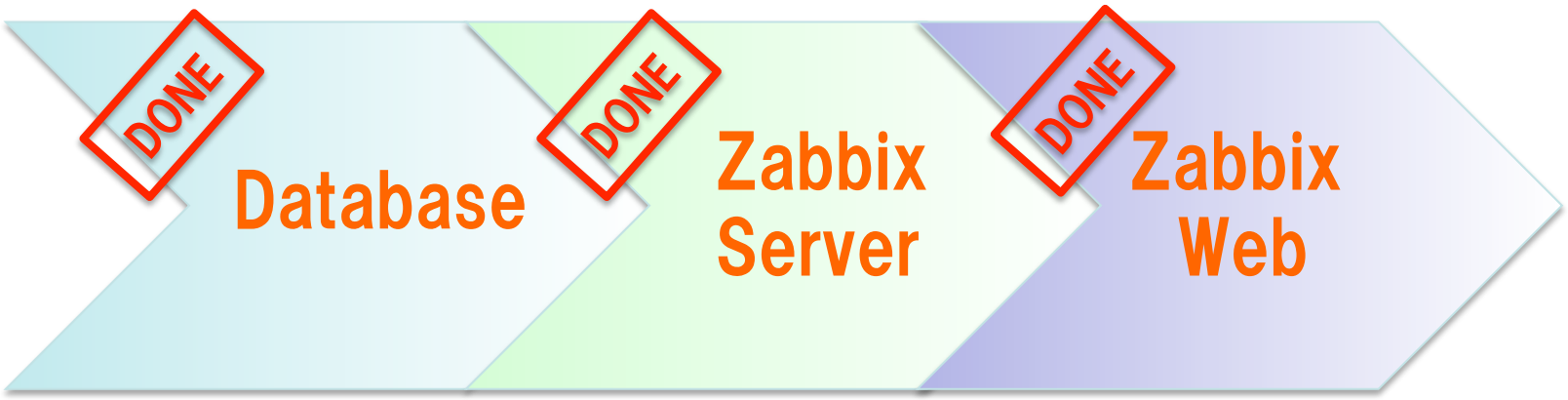
- user.create
  - severity によって通知先の Email アドレスを分けておく
- mediatype.create
  - 通知 Email の SMTP サーバーの設定等
- action.create
  - 実際に送信されるメール、リカバリメールの文面
  - リカバリメッセージの送信設定
  - アクションの実行条件（メンテナンス期間外、トリガーの値=”障害”）
- template.delete
  - 初期 template の削除
- configuration.import
  - オリジナル template の import

Chef による Zabbix 監視自動化について / API user.create

```
{
    "auth": " 1234567890abcdefghijklmnopqrstu",
    "id": 1234,
    "jsonrpc": "2.0",
    "method": "user.create",
    "params": {
        "alias": " zabbix_account",
        "name": " zabbix_account",
        "passwd": " zabbix_password",
        "surname": " zabbix_account",
        "type": 3,
        "user_medias": [
            {
                "active": 0,
                "mediatypeid": 1,
                "period": "1-7,00:00-24:00",
                "sendto": " crit@makocchi.com",
                "severity": 56
            },
            {
                "active": 0,
                "mediatypeid": 1,
                "period": "1-7,00:00-24:00",
                "sendto": " warn@makocchi.com",
                "severity": 63
            }
        ],
        "usrgrps": [
            {
                "usrgrpid": "0"
            }
        ]
    }
}
```

例えば user.create だと
こんな感じの json を API で渡してあげます

Zabbix Server を語る上で必要なコンポーネントは 3 つ



いろいろ話してきましたが、今までのスライドで書いてきたことが
コマンド 1 発 (chef-client) で入ります

サーバー起動時の KickStart に chef-client を仕込んでおけば
コマンドを叩く必要さえありません
ログインしなくても zabbix server ready

Zabbix Agent のセットアップ

```
# chef-solo -o zabbix::agent_setup
```

Zabbix Agent の recipe でやることは下記の通り

1. パッケージインストール
   - 例によって Zabbix 公式の rpm (repo.zabbix.com)

2. /etc/zabbix/zabbix_agentd.conf の編集
   - UnsafeUserParameters=1 (必要に応じて)
   - Hostnameitem ではなく Hostname=FQDN

3. Zabbix Server 側へ API で監視登録
   - 詳細は次ページ

Zabbix Server 側へ下記の流れで API を叩きます

1. hostgroup.create
   - hostgroup が無いと host.create できないので
2. host.create
   - ipaddress, fqdn を登録（ohaiより自動取得）
3. hostinterface.create
   - Jmx, SNMP, IPMI を必要に応じて設定
4. template.massadd
   - インストールするミドルウェアに応じた template を投入

ちょっと chef 的なお話

- インストールするミドルウェアに応じて設定する template を変える
  例えば apache がインストールされたら apache 用の template を、
  nginx がインストールされたら nginx の template を適用する
- 各種ミドルウェアのインストールも chef の recipe によって行われる
- その recipe に設定されるべき template を記述しておけばよい

ちょっと chef 的なお話2

- 具体的に書いてしまうと

```
run_list: [
  recipe[zabbix::agent]
  recipe[apache]
]
```

- zabbix::agent 内で node.set しておく

```
node.set [:zabbix] [:agent] [:enable] = true
```

- apache の recipe 側でハンドリングする

```
if node[:zabbix][:agent][:enable]?
  Chef::Log::info('use " zabbix" for monitoring.')
  include_recipe  "apache::zabbix"
end
```

ちょっと chef 的なお話3

- apache::zabbix には設定すべき template 名と API の処理を書いておく

```
…

zabbix_host node[:zabbix][:api][:server] do
  templates [ “apache” ]
  action :create
End
```

API 処理

- 全ての system で zabbix を使うわけではないので、zabbix::agent を適用させるサーバーのみ zabbix api が叩かれるようにしておく

ちょっと chef 的なお話4

- こんな感じだと・・・

```
run_list: [
  recipe[zabbix::agent]
  recipe[nginx]
  recipe[memcached]
  recipe[mysql]
]
```

nginx, memcached, mysql の template が設定される

- 実は zabbix 以外もできるようにしてあったり

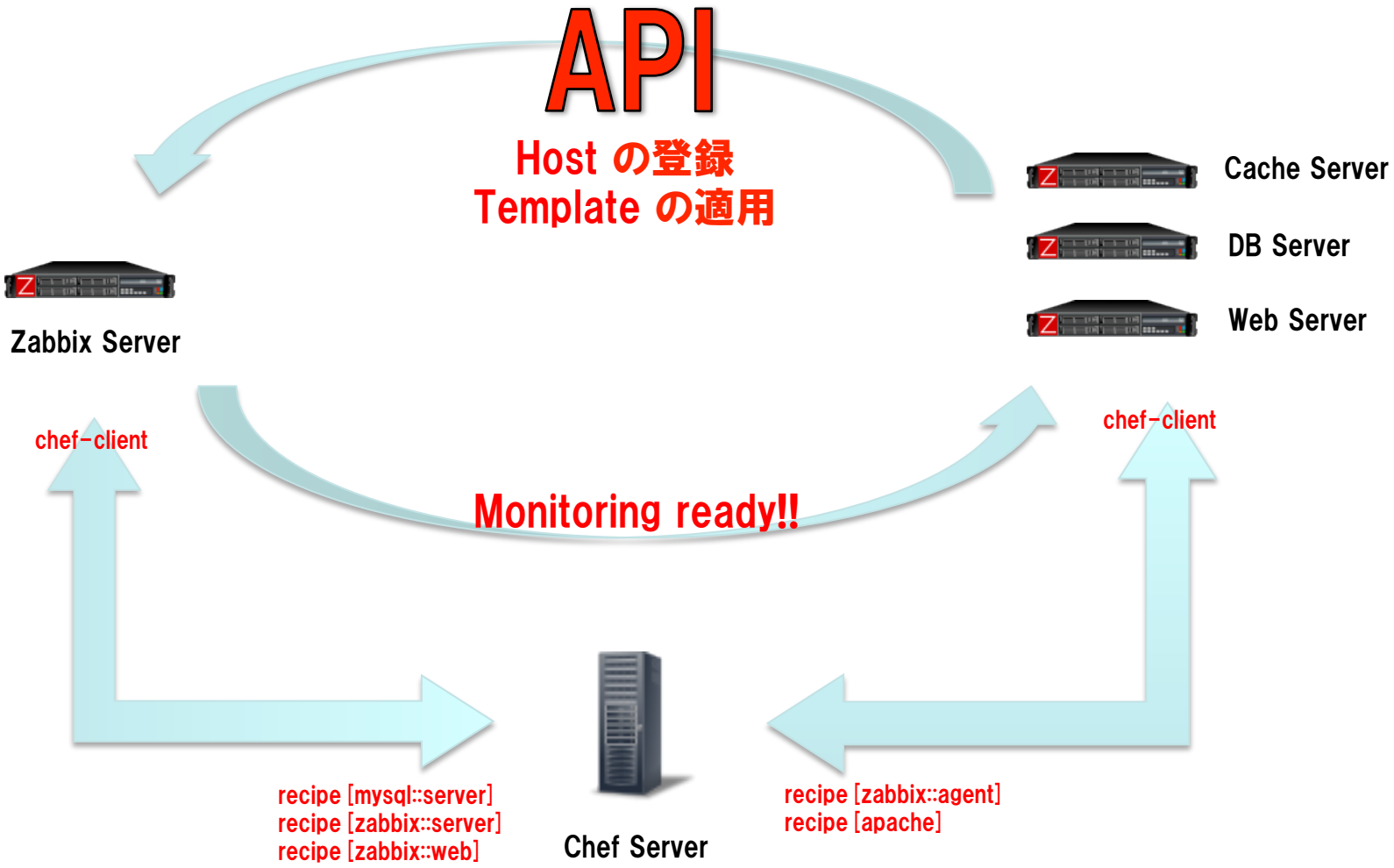
```
run_list: [
  recipe[nagios::agent]
  recipe[nginx]
  recipe[memcached]
  recipe[mysql]
]
```

```
run_list: [
  recipe[munin::client]
  recipe[nginx]
  recipe[memcached]
  recipe[mysql]
]
```

おさらい
図にしてみます

```
# chef-solo -o summary
```

Chef による Zabbix 監視自動化について / 図



おわりに

```
# chef-solo -o closing
```

Template の管理と deploy

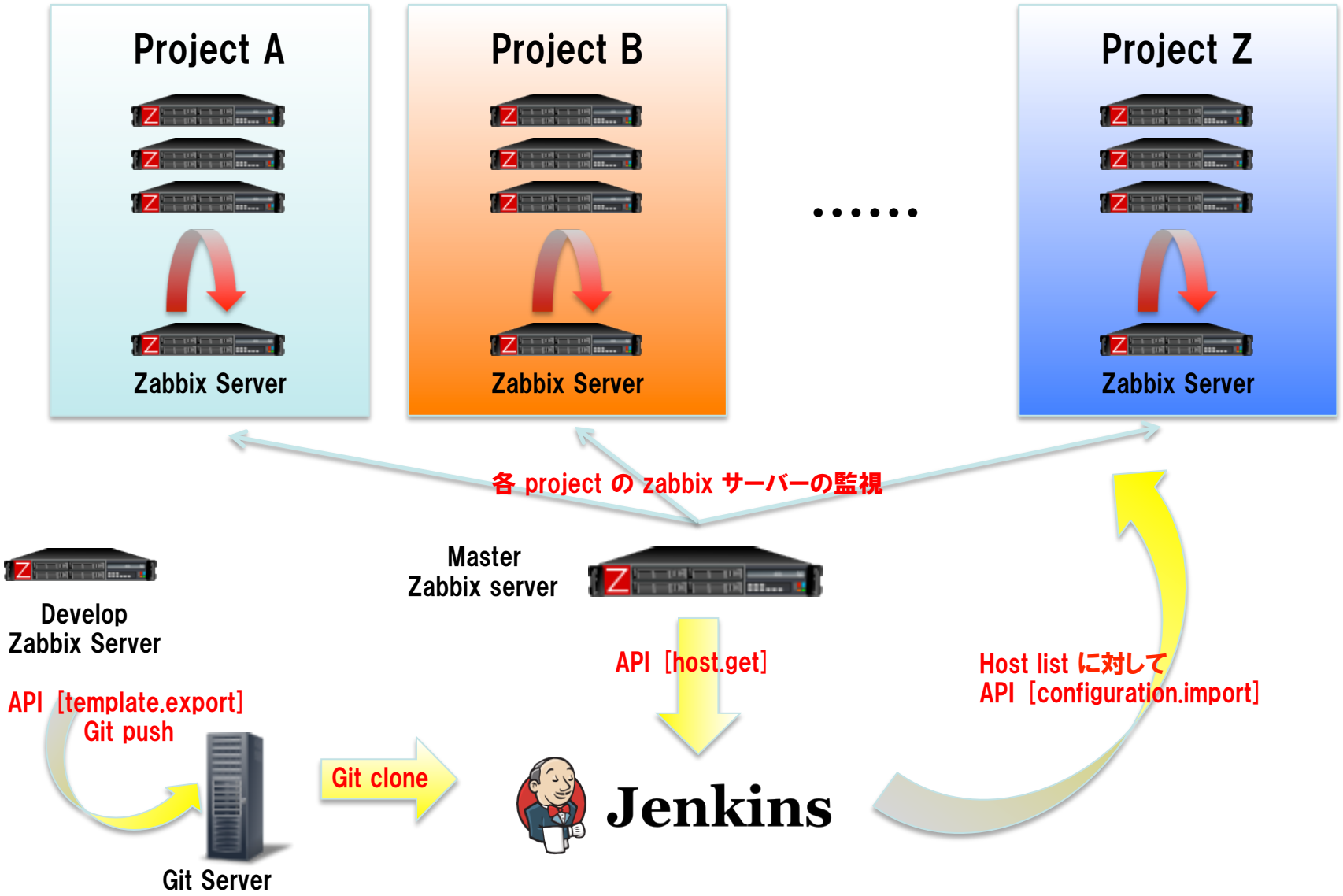
```
# chef-solo -c extra.rb -o zabbix::template_management
```

Zabbix の template 管理はどうされていますでしょうか？

- export した xml を git や svn で管理？
- git や svn の xml を既存の zabbix にどうやって deploy する？
- 管理用 zabbix がいてそこで template をいじったりしている？
- してないよ　そんなもん

あくまで1例ですが、
弊社ではどうしているかというと・・・

Chef による Zabbix 監視自動化について / Ex template management



Classic style からの脱却

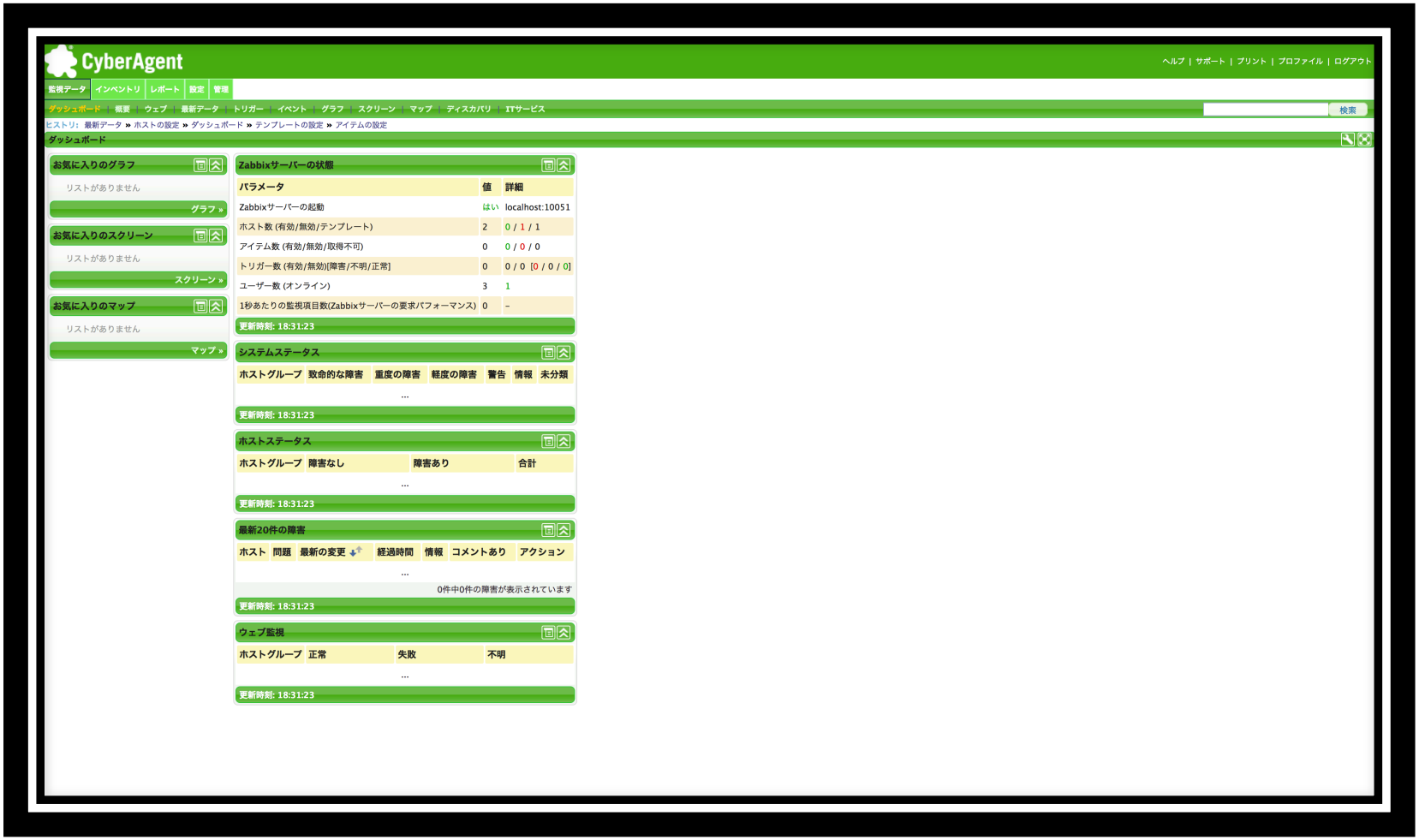
```
# chef-solo -c extra.rb -o zabbix::custom_style
```

Zabbix にはデフォルトでいくつか Theme が入っています

Classic / Dark orange / Black & Blue …

せっかくだから新しい Theme 作ってみようかな・・

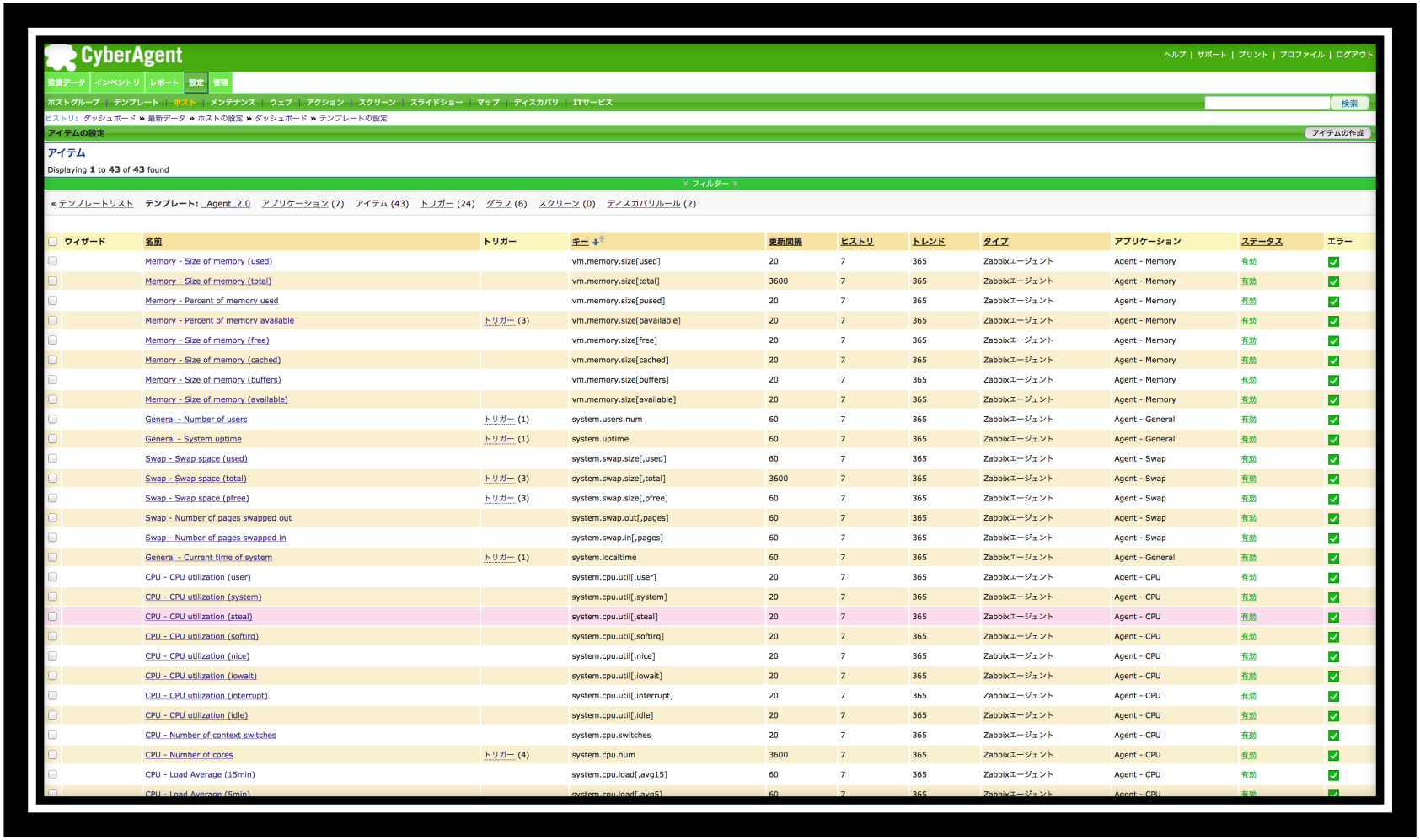
Chef による Zabbix 監視自動化について / Ex custom_style



こんな感じで Ameba 色にしてみました

Chef による Zabbix 監視自動化について / Ex custom_style

こんな感じで Ameba 色にしてみました2



ウィザード	名前	トリガー	キー	更新間隔	ヒストリ	トレンド	タイプ	アプリケーション	ステータス	エラー
	Memory - Size of memory (used)		vm.memory.size[used]	20	7	365	Zabbixエージェント	Agent - Memory	有効	
	Memory - Size of memory (total)		vm.memory.size[total]	3600	7	365	Zabbixエージェント	Agent - Memory	有効	
	Memory - Percent of memory used		vm.memory.size[pused]	20	7	365	Zabbixエージェント	Agent - Memory	有効	
	Memory - Percent of memory available	トリガー (3)	vm.memory.size[pavailable]	20	7	365	Zabbixエージェント	Agent - Memory	有効	
	Memory - Size of memory (free)		vm.memory.size[free]	20	7	365	Zabbixエージェント	Agent - Memory	有効	
	Memory - Size of memory (cached)		vm.memory.size[cached]	20	7	365	Zabbixエージェント	Agent - Memory	有効	
	Memory - Size of memory (buffers)		vm.memory.size[buffers]	20	7	365	Zabbixエージェント	Agent - Memory	有効	
	Memory - Size of memory (available)		vm.memory.size[available]	20	7	365	Zabbixエージェント	Agent - Memory	有効	
	General - Number of users	トリガー (1)	system.users.num	60	7	365	Zabbixエージェント	Agent - General	有効	
	General - System uptime	トリガー (1)	system.uptime	60	7	365	Zabbixエージェント	Agent - General	有効	
	Swap - Swap space (used)		system.swap.size[,used]	60	7	365	Zabbixエージェント	Agent - Swap	有効	
	Swap - Swap space (total)	トリガー (3)	system.swap.size[,total]	3600	7	365	Zabbixエージェント	Agent - Swap	有効	
	Swap - Swap space (pfree)	トリガー (3)	system.swap.size[,pfree]	60	7	365	Zabbixエージェント	Agent - Swap	有効	
	Swap - Number of pages swapped out		system.swap.out[,pages]	60	7	365	Zabbixエージェント	Agent - Swap	有効	
	Swap - Number of pages swapped in		system.swap.in[,pages]	60	7	365	Zabbixエージェント	Agent - Swap	有効	
	General - Current time of system	トリガー (1)	system.localtime	60	7	365	Zabbixエージェント	Agent - General	有効	
	CPU - CPU utilization (user)		system.cpu.util[,user]	20	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - CPU utilization (system)		system.cpu.util[,system]	20	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - CPU utilization (steal)		system.cpu.util[,steal]	20	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - CPU utilization (softirq)		system.cpu.util[,softirq]	20	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - CPU utilization (nice)		system.cpu.util[,nice]	20	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - CPU utilization (iowait)		system.cpu.util[,iowait]	20	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - CPU utilization (interrupt)		system.cpu.util[,interrupt]	20	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - CPU utilization (idle)		system.cpu.util[,idle]	20	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - Number of context switches		system.cpu.switches	20	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - Number of cores	トリガー (4)	system.cpu.num	3600	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - Load Average (15min)		system.cpu.load[,avg15]	60	7	365	Zabbixエージェント	Agent - CPU	有効	
	CPU - Load Average (5min)		system.cpu.load[,avg5]	60	7	365	Zabbixエージェント	Agent - CPU	有効	